CA Release Automation 5.5

Architecture and Implementation

Best Practice Guide

Author :	Keith Puzey
Version:	2.6
Filename:	CA Release Automation Architecture and Implementation Guide 5.5
Date:	Thursday, 21 January 2016

Table of Contents

Document History

Version	Date	Description
1.6	July 4th 2014	Initial Document Released
1.7	July 7th 2014	Tuning Tips section Added Monitoring Section Added
1.9	September 1st 2014	Updated STAR Execution server section
2.0	October 21st 2014	Included SSL ports for ActiveMQ in network section Updated Version table to Include SP2 Updated Tuning Section Updated Appendix A details
2.1	January 8th 2015	Release Automation 5.5 update
2.2	January 16th 2015	Updated Tuning Section Legal Notice Added Updated section on Artifact Distribution
2.3	February 18th 2015	Update service section updated.
2.4	July 28th 2015	Build History table updated URL for System Recommendations Updated Update to Execution Server architecture section to include ring architecture
2.5	December 14th 2015	Updated build history Update to Execution Server routing architecture
2.6	January 19th 2016	General reformatting of document to consistently use Execution server instead of NES Moved file configuration details to new Appendix

CA Release Automation Components

Database Component

CA Release Automation supported Database servers:

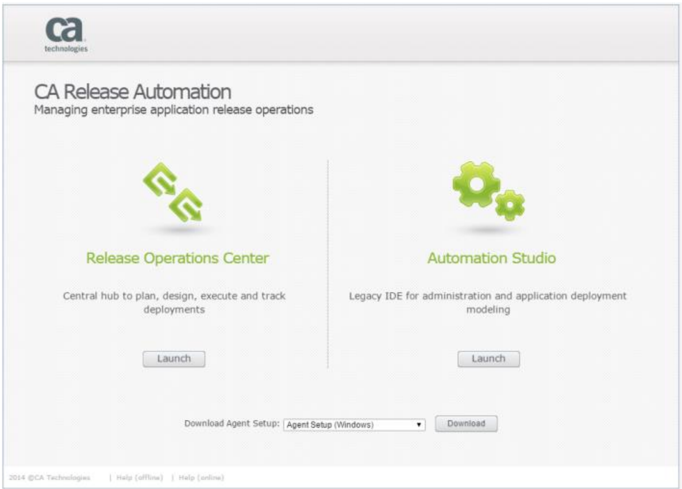
MySQL Version 5.1.50 to 5.6

Oracle Version 10g to 12c

Microsoft SQL Server Version 2008 to 2012 SP1

Management Server (NAC)

The Management Server is the central point of a "CA Release Automation" deployment it has several functions which include the scheduling and process control, when a process is executed within the management server the relevant instructions are sent to the agents via the Execution Servers that are assigned to the agents. The Management server also includes the UI server, when connecting to the Management Server with a browser the landing page has launch points to the two UI's:



CA Release Automation
Managing enterprise application release operations

Release Operations Center

Central hub to plan, design, execute and track deployments

Automation Studio

Legacy IDE for administration and application deployment modeling

Download Agent Setup: Agent Setup (Windows)

2014 ©CA Technologies | Help (offline) | Help (online)

Release Operation Center – ROC, Management Server

As the controller of the system, the Management Server drives all executions, manipulates data in the database, calculates and updates flows states. The Management Server also temporarily holds all files that are needed for currently started executions.

Automation Studio (ASAP)

IDE for generic application deployment modelling across multiple tiers. When launching the Automation Studio interface a Java Web Start application is downloaded to the local machine (Java JRE is required). The Java web start ensures that the latest client will be installed and updated.

With the introduction of CA Release Automation 5.5 the designer functionality is being migrated to the new designer functionality in Release Operations Center. The administration functions will be migrated from the Automation studio in a future release.

Release Automation Repository

The repository provides storage for artifacts to be used in Release Operations Center releases and for all action packs (core and custom) to be used in processes. The repository server (Nexus) is installed by default as part of the Management server installation. The installation of the Management Server includes an option to connect to an external repository.

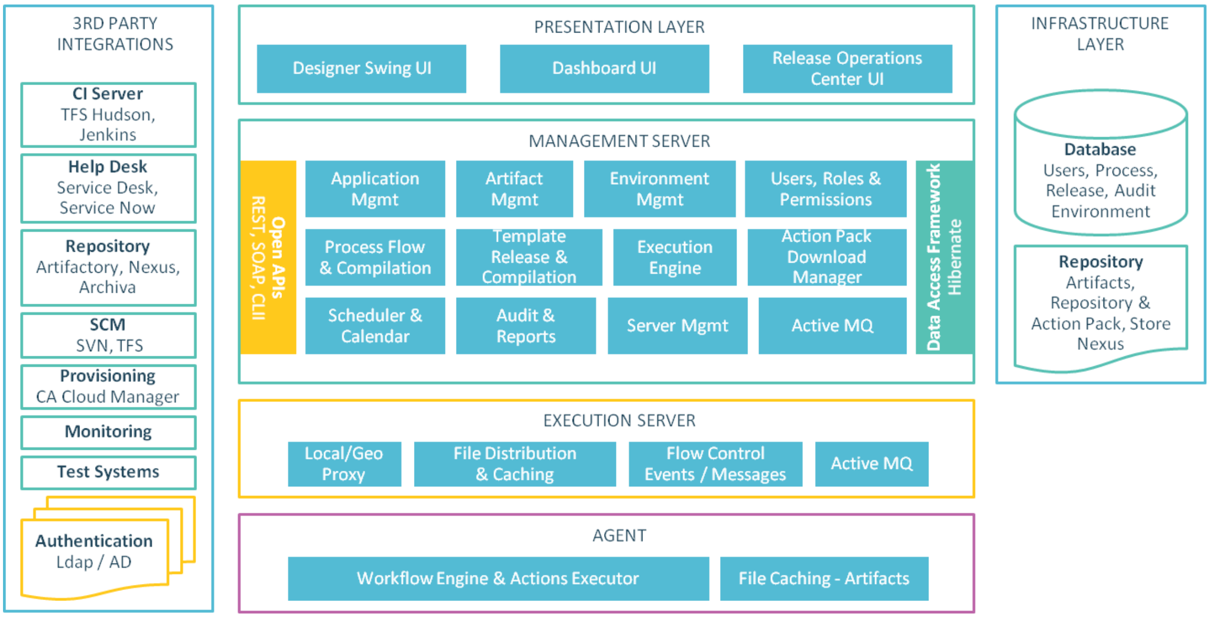
Execution Server (NES)

A CA Release Automation deployment must have at least one Execution server, The Execution Server mediates between the Management Server and the Agents. The Execution Server distributes control messages and files to the Agents and collects status data from the Agents which is retrieved later by Management Server. The Execution Server serves as a bridge when an agent needs to communicate with other agents and a direct route is not available. More than one Execution Server can participate in the communication between two agents.

Agents (AGT)

All Machines that are part of the CA Release Automation environment require an Agent to be installed. Agents can be remotely deployed from the Automation studio or installed manually. Agents are the main workers of the system, each Agent is deployed on a server which it manages. The Agent gets the process flow and files from the Execution Server and sends process execution status back to the Execution Server. Agents can also be used to retrieve artifacts from the Repository.

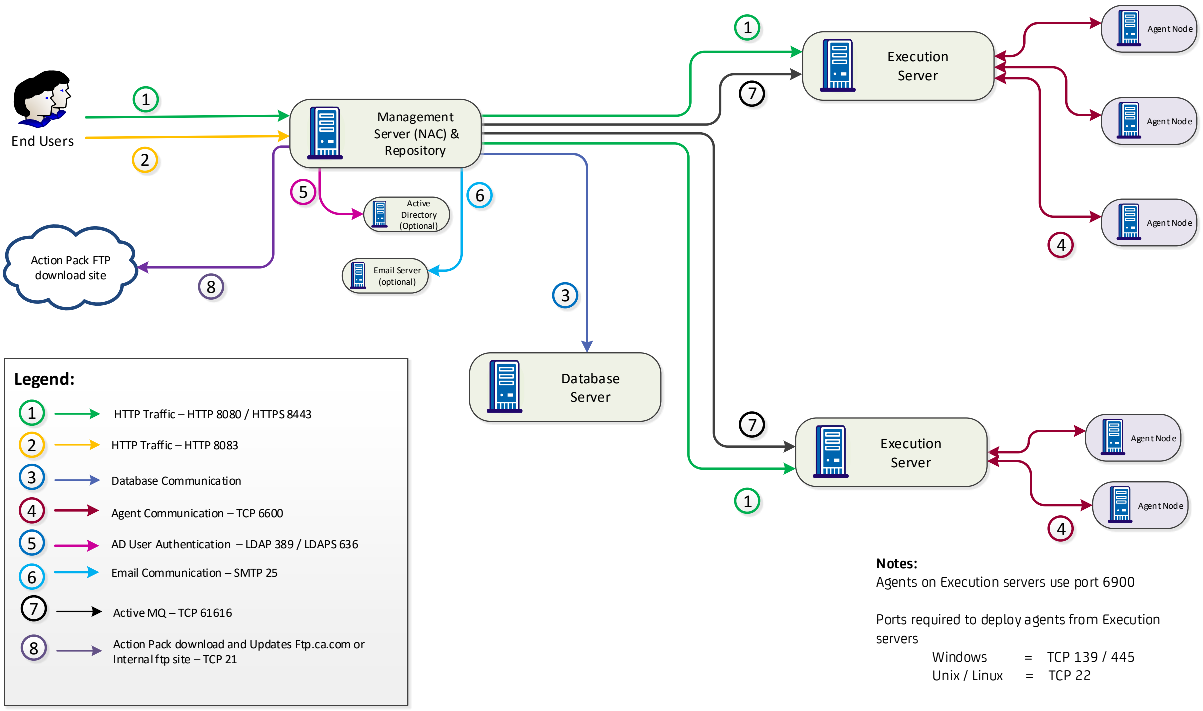
Logical Architecture



Release Automation 5.x Revision History

Release Date	Version	Description
18th March 2014	5.0.617	GA build 5.0 – Build 617
11th June 2014	5.0.1.764	5.0 Service Pack 1 – build 764
29th September 2014	5.0.2.78	5.0 Server Pack 2 – Build 78
24th December 2014	5.5.0.831	GA Build 5.5.0 - Build 831
15th January 2015	5.5.0.842	Hot Fix – 5.5.0 - Build 842
28th January 2015	5.5.0.849	Hot Fix – 5.5.0 - Build 849
5th May 2015	5.5.1.1312	5.5 Service Pack 1 - Build 1312
14th July 2015	5.5.2.191	5.5 Service Pack 2 - Build 191
29th October 2015	5.5.2.218	Hot Fix - 5.5.2 Build 218
25th November 2015	5.5.2.236	Hot Fix - 5.5.2 Build 236
11th December 2015	5.5.2.239	Hot Fix - 5.5.2 Build 239

High Level Component and Port Architecture



Legend:

1	HTTP Traffic – HTTP 8080 / HTTPS 8443
2	HTTP Traffic – HTTP 8083
3	Database Communication
4	Agent Communication – TCP 6600
5	AD User Authentication – LDAP 389 / LDAPS 636
6	Email Communication – SMTP 25
7	Active MQ – TCP 61616
8	Action Pack download and Updates Ftp.ca.com or Internal ftp site – TCP 21

Notes:
Agents on Execution servers use port 6900

Ports required to deploy agents from Execution servers

Windows	=	TCP 139 / 445
Unix / Linux	=	TCP 22

The schematic shows the various components that make up a typical CA Release Automation deployment.

Example 5.x Deployments

The following table shows some examples of Release Automation deployments, the first two examples are of complex networks where execution servers are deployed in multiple remote data centers. Details on the he minimum hardware requirements for release automation can be found on the CA Wiki at the following URL: https://wiki.ca.com/display/RA55/System+Requirements

Overview	Database Server	NAC Server	Execution Server
2 NAC 12 x Execution Server 1300 x Agent's 23 Application's 4000 Deployments / Month	Linux RedHat 5.8 Oracle DB 11G R2 Physical 12 Core / 64 GB Ram Disk 256 Gb	Linux RedHat 5.8 Virtual (Xen) 8 vCpu's / 32 GB Memory Disk - 256 Gb	Linux RedHat 5.8 Virtual (Xen) 4Vcpu's 32 Gb Memory Disk 128 Gb Maximum 350 Agents
2 x NAC's 22 x Execution Server 1700 x Agents 25 Applications	Windows 2008 R2 SP2 Enterprise Clustered MS SQL 2008 R2 Standard Physical HP BL460 G6 8 Cores, 24gb Memory Disk – 350Gb Database 220Gb	SLES 11 Physical HP BL460 G6 8 Core, 12gb Memory	SLES 11 Physical IBM B4, 8 Core, 16Gb Memory Virtual 4vCPU, 16gb Memory Disk -50-300Gb Maximum 350 Agents
1 x NAC 2 x Execution Server 218 x Agents 3 Applications 25 Deployments per week	Unix Server –Virtual (the database server is one of 6 VMS on the underlying physical server) Oracle DB 11G R2 CPU Threads: 28 Database Size 45Gb	Windows 2008 R2 Physical, Dual Processor Quad Core Hyper-threaded, 8GB Memory Disk - 99Gb and 738Gb	Windows 2008 R2 64 Bit Physical, Dual Processor Quad Core Hyper-threaded 8GB Memory Disk 715Gb Maximum 218 Agents

Architecture Guidelines

The following section contains information that should be considered when sizing Release Automation instances (NAC's) and when additional NAC's may be required:

- A single Release Automation instance can support several thousand agents.
- A second NAC may be required when a large number of teams are sharing the same instance teams can potentially have different release cycles so it can be very hard to schedule a maintenance window for the Release Automation Server itself. For example when an upgrade to the latest Release Automation version is required, it can be very hard to find a consistent time that is suitable for all users. By limiting the number of teams per NAC, each group has the ability to schedule upgrade activities around project commitments and increase the chance of planning a maintenance window.
- When a project wants to define its environments in Release Automation, the user may have to navigate through a very large list of irrelevant agents. There are features in Release Automation to assist with this, e.g. .the ability to group agents in to a custom folder structure, but having a smaller set of agents (e.g. Hundreds to thousands rather than thousands to tens of thousands) per NAC improves the general manageability.
- User and permissions management is simpler. This reduces the risk of incorrect configuration and users having access to agents and processes they should not have access to.
- Log file s are easier to navigate without the additional messages created by all the additional agents. With this in mind then, we recommend planning to not have 1 single NAC for an entire estate of tens of thousands of servers, but instead plan to have a NAC per related business area. A related business area is one in which the use of a Release automation installation is on a similar / shared area, and where the users can plan required maintenance windows as discussed above.
- The exact number of agents that can connect to an Execution Server depends greatly on many factors, including:

  - Number of executions performed in parallel

  - The complexity of those executions

  - The size of files transferred during the process executions

As such it is very difficult to accurately predict the limit. As a rule of thumb we normally advise that one Execution Server can handle a maximum of 1500 agents concurrently and no more than 400 agents should be active at the same time.

We typically find that a Release Automation implementation is not rolled out to all servers in one go, and that instead the agents are rolled out gradually over a period of time, team by team and environment by environment. In this way, it is possible to monitor the performance of the Execution Server and verify workload with the demand that is being placed upon it. Should it be decided to include the Release Automation agent as part of the standard build on all servers, this is possible but we would recommend that the Release Automation service is installed in a latent state allowing phased roll out of Release Automation by simply activating the Release Automation agent services as the agents are brought in to use.

Execution Servers should be placed electronically close to agent machines where ever possible.

Another factor which can affect the number and location of Execution Servers, is the network design and in particular the security and geographical aspects. If the NAC is not physically located with the target servers (e.g.it is in a separate datacentre), it is a good idea to place an Execution Server near to those physical servers (see the Network Latency considerations section). This allows the NAC to talk in an efficient manner with the more remote Execution Server, and the Execution Server can then have a higher volume of local communication with the many target servers. In terms of network security, a single Execution Server can manage multiple environments including pre---production and production, but it may be necessary to consider separating some environments, e.g. Production, from the others---either from a pure network security perspective (maybe there is not one location that can see both pre---production and production) or from a risk and compliance viewpoint.

When using a repository / utility agent for artifact retrieval the best practice is to use “Artifact Retrieval” groups containing multiple agents. If an application relies on a movement of a large number of artifacts it is recommended to use separate “Artifact Retrieval” agents. The retrieval agents should be placed electronically close to the source of the artifacts.

Deployments will generally include a mixture of single Execution Servers and Super Execution Servers, a super Execution Server should be used where resilience is required such as production environment where as a test or QA environment may only require a single Execution Server.

The effect of network latency varies depending upon which Release Automation components experience the latency. Between the NAC and Execution Server, and between Execution Server and Agent, latency can have the effect of increasing the overall execution time of a deployment process. This would be particularly noticeable if the process is transferring very large files across the affected connections. For this reason some customers have separated the file distribution aspects of a deployment process into one Release Automation process, and the actual deployment activities into a second process. The distribution process can then be executed at a convenient time before the deployment is actually required. If the latency is between the Release Automation client UI and the NAC, it can have different detrimental effects on the user, e.g. the time to load the client UI (as the UI has to load all the relevant applications, components, environment information etc.), and the time it takes to perform all actions (such as adding an action, deleting an action, changing a parameter value etc.). Latency can be reduced by taking into consideration the location of users and NACs and providing appropriate network connectivity between these locations.

For production deployment it is recommended to use Microsoft SQL or Oracle Databases. The database is used to store all the data relating to your use of the product including:

a) The processes themselves
b) The process parameters and their values
c) The environment definitions
d) Process execution results
e) Administration settings such as user accounts, LDAP configuration, permissions settings etc.

As such the amount of data created is heavily dependent upon a number of factors including the size and complexity of the processes, the number of executions performed in a given time period etc. As such we are not able to accurately predict the database growth until we can gather real usage metrics.

Monitoring Health and Availability

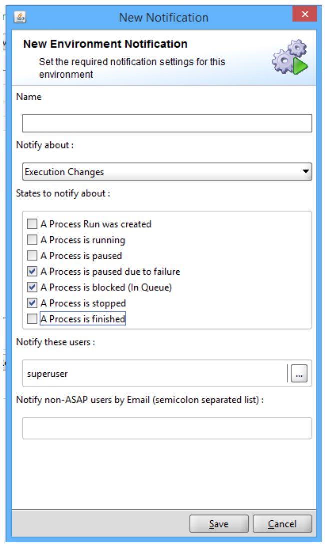
CA Recommends that to ensure that CA Release Automation is functioning at its optimum the operating systems hosting the application servers are monitored using a system monitoring tool, the table below shows the recommended monitoring for each application server type.

Application Server	**Metric**	**Description**
Database	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where the Release Automation Database is located to ensure that plenty of disk capacity is available.
NAC	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.

	Windows Service	Ensure that the following windows service is running:
		ServiceName = NolioServer20
		ServiceName = Nolio Update Service

	Log file	nolio_dm_all.log - Monitor for a string “error “
Repository	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.
	Windows Service	Ensure that the following windows service is running: ServiceName = RepoService
Execution Server	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.
	Windows Service	Ensure that the following windows service is running: ServiceName = NolioServer20
	Log file	nolio_exec_all.log - Monitor for a string “error “

As well as monitoring the operating system it is important for the CA Release Automation administer to be notified if any issues are being experienced with jobs/ processes. The simplest way to be notified of these events is to configure the SMTP server settings in the ASAP "System Settings" and to create Notification policies for each environment. The Status's that an administrator would find useful to be notified of are:



New Environment Notification

Set the required notification settings for this environment

Name

Notify about : Execution Changes

States to notify about :

- □ A Process Run was created
- □ A Process is running
- □ A Process is paused
- ☑ A Process is paused due to failure
- ☑ A Process is blocked (In Queue)
- ☑ A Process is stopped
- □ A Process is finished

Notify these users : superuser

Notify non-ASAP users by Email (semicolon separated list) :

"A Process is Paused due to failure" – Diagnose Paused Process and identify the failing part of the process.

"A Process is blocked (In Queue)" - A Process has been Queued due to an agents which is required not being available. This is normally caused by another process being paused or failed which is blocking the agent.

"A Process is stopped"

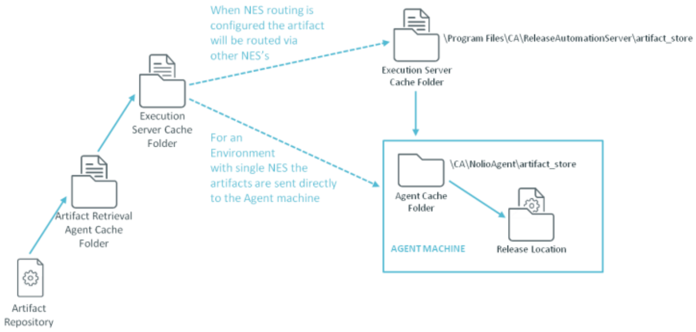
Backup and Recovery Recommendations

When creating a Backup procedure for CA Release Automation the database is the most important component that required regular backups. It is recommended to complete a full Database backup weekly and incremental backup between these full back ups if possible backups should be taken when the system has no running jobs. If the CA Release Automation Nexus Repository is being used to store artifacts this should be backed up in the same way as the Database. The NAC / Repository and Execution Server installation folders should be backed up after any major upgrades

Automated artifact Distribution

Execution servers communicate directly with agents and should be placed electronically close to the agents being managed. As well as the physical placement of Execution servers there is also a logical connectivity of the Execution Servers which is used for routing messages and files between agents.

When release automation distributes artifacts the artifacts are automatically routed via the execution servers using the Execution Server routing configuration which is covered in the following sections, the following graphic shows how artifacts are routed.


When NES routing is configured the artifact will be routed via other NES's
\Program Files\CA\ReleaseAutomationServer\artifact_store
Execution Server Cache Folder
Execution Server Cache Folder
For an Environment with single NES the artifacts are sent directly to the Agent machine
\CA\NolioAgent\artifact_store
Agent Cache Folder
Artifact Retrieval Agent Cache Folder
AGENT MACHINE
Release Location
Artifact Repository

A more complete explanation of the artifact distribution process has been published on the CA Community site https://communities.ca.com/docs/DOC-231151475 and the contents of the knowledge base article have been included in this document.

The process of artifact distribution and cleanup has several stages, which we will outline below.

Artifact Execution Server Distribution

1. Artifacts are retrieved by retrieval agent and downloaded to <Retrieval_Agent_Root>/files/ART<Release_ID>
2. They are moved to <Retrieval_Agent_Root>\files_registry\<File_MD5>
   - This sub-stage is sometimes referred to as the "Retrieval Stage". While the files are being moved, there could be a point where the file exists on the agent at both locations at the same time (thus peaking at two times its size). After the move finishes, they exist only in 'files_registry'.
3. They are copied to the Execution Server at <NES_Root>\artifact_store\releaseId_<Release_ID >
   - Each Execution Server will request and get only the artifacts that are needed by the agents that are connected to it.
4. They are deleted from <Retrieval_Agent_Root>\files_registry\<File_MD5>.

The files now reside exclusively on <NES_Root>\artifact_store\releaseId_<Release_ID>

Artifact Agent Distribution

1. All the release artifacts are now copied to the Execution Server, at <NES_Root>\files_registry\<File_MD5>
   - At this point the file exists on both locations in the Execution Server : 'files_registry' and 'artifact_store'.
2. Every agent requests the artifacts it needs from its Execution Server and downloads them to a folder named: <Agent_Root>\artifact_store\releaseId_<Release_ID>
   - The artifact will remain available there during the release processes runs.
3. After all the agents have finished downloading the artifacts from their Execution Servers, the artifacts will be deleted from <NES_Root>\files_registry\<File_MD5>

The files now reside on two locations:

<NES_Root>\artifact_store\releaseId_<Release_ID>

<Agent_Root>\artifact_store\releaseId_<Release_ID>

Artifact Cleanup

When the release reaches a final state ("failed" or "finished"), the NAC sends an order to all the Execution Servers and agents participating in the release to delete the folders named with the release ID under the 'artifact_store' folder.

This should clear all the disk space retained by the release artifacts.

However, it is possible that some of the Execution Servers/agents weren't reachable at the end of the release and have missed the cleanup command sent from the NAC.

In order to avoid keeping obsolete artifacts, once a day each Execution Server and agent checks if all the release IDs under the 'artifact_store' folder are still relevant (i.e. the release in not in a final state) and deletes all the obsolete ones.

The release is finished and a day has passed.

The files now will not exist on the retrieval agent, ExecutionServer or target agent machines.

Artifact Caching

Every time a node transfers a file using an agent, it will also be stored in this node cache and used later for optimizations.

From the RA perspective the disk space allocated for caching (1 GB by default) is always considered as used by the system (even if it can be sometimes be free).

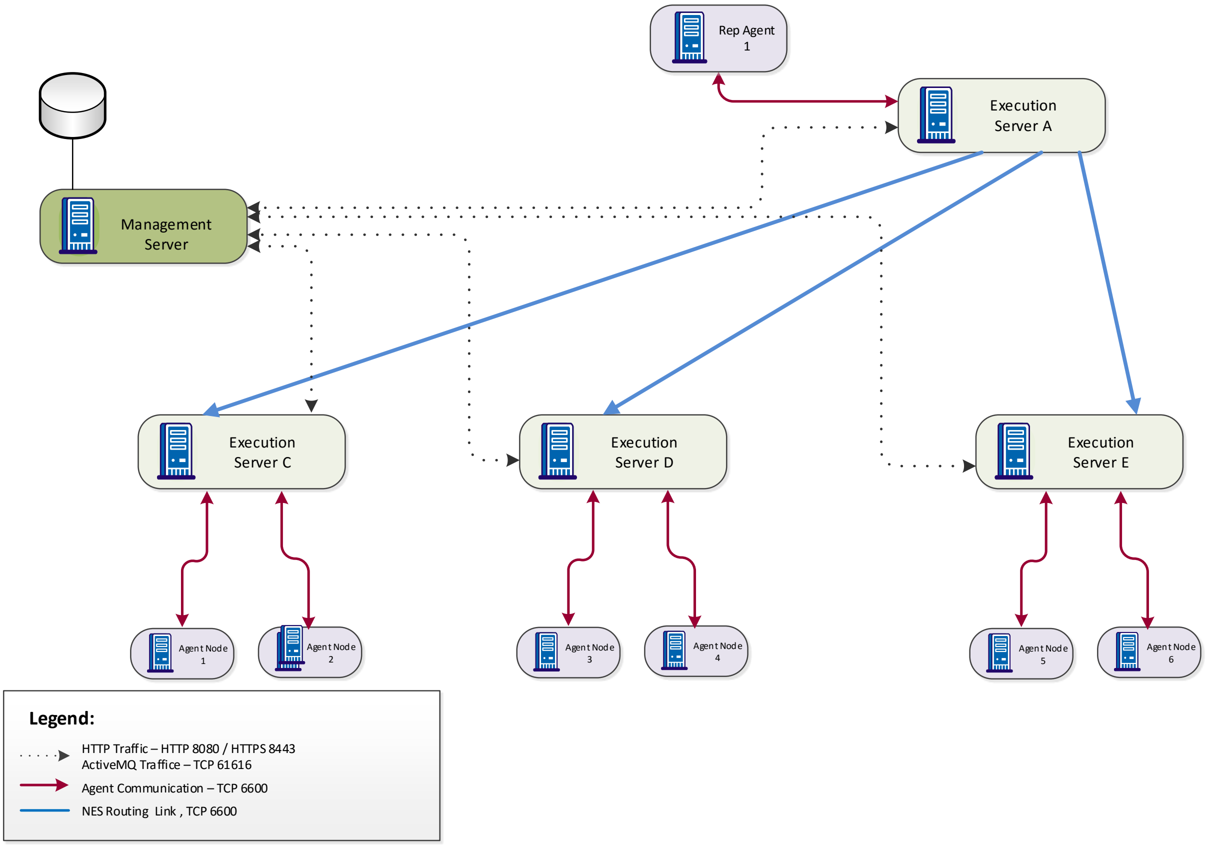
The cache works in LRU mode by default and will discard the least recently used file first, when it reaches the size limit.

The following section contains examples of different Execution Server architectures.

STAR Execution Server Routing Architecture

In an architecture involving multiple datacentre's or environments the execution servers are installed within the datacentre's or separated environments, in this architecture a process can only copy files between agents attached to the same execution server. If the processes require access to other agents within Release Automation the execution servers can be connected in a STAR configuration as seen in the following schematic. Connecting the Execution servers in this logical arrangement allows any agent to copy files to any other agent. As an example if "Agent Node 1" is required to copy a file to "Agent Node 6", "Agent Node 1" will send a request to "Execution Server C" to look up the location of "Agent Node 6" as the Execution server does not have a connection to "Agent Node 6" the request will be passed to "Execution Server A", this execution server also does not have a connection to "Agent Node 6" and the request will be passed to "Execution Server E" as "Agent Node 6" is connected to "Execution Server E" this information is passed to "Agent Node1" and a route is defined between the two agents.

The blue lines indicate the execution server inter connections which are used to transfer artifacts between agents / execution servers. The best practice when creating this Execution Server routing architecture is to define the connection from the central execution server as shown by the arrows in the drawing, the connection is initiated from the source execution server but once the connection is made data flows in both directions.

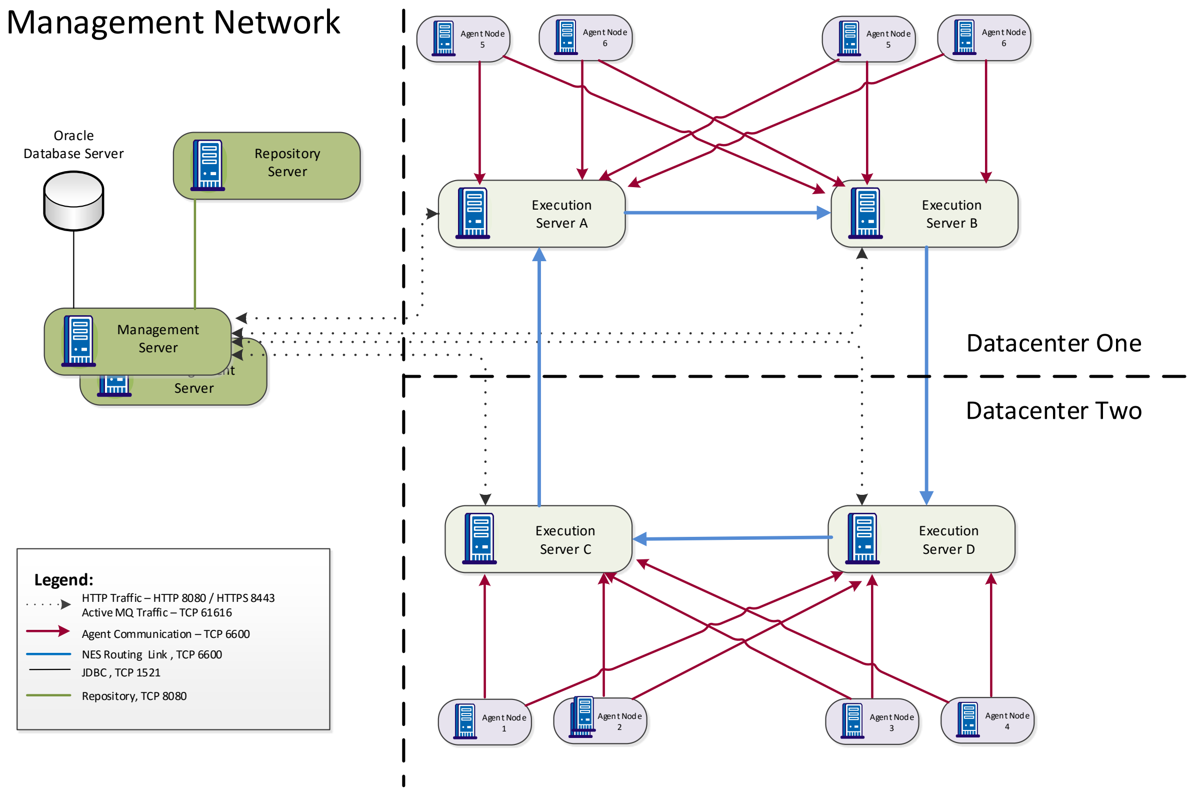


Note: Before configuring a STAR architecture ensure that the environment is upgraded to the latest maintenance build.

Ring Execution Server with High Availability Architecture

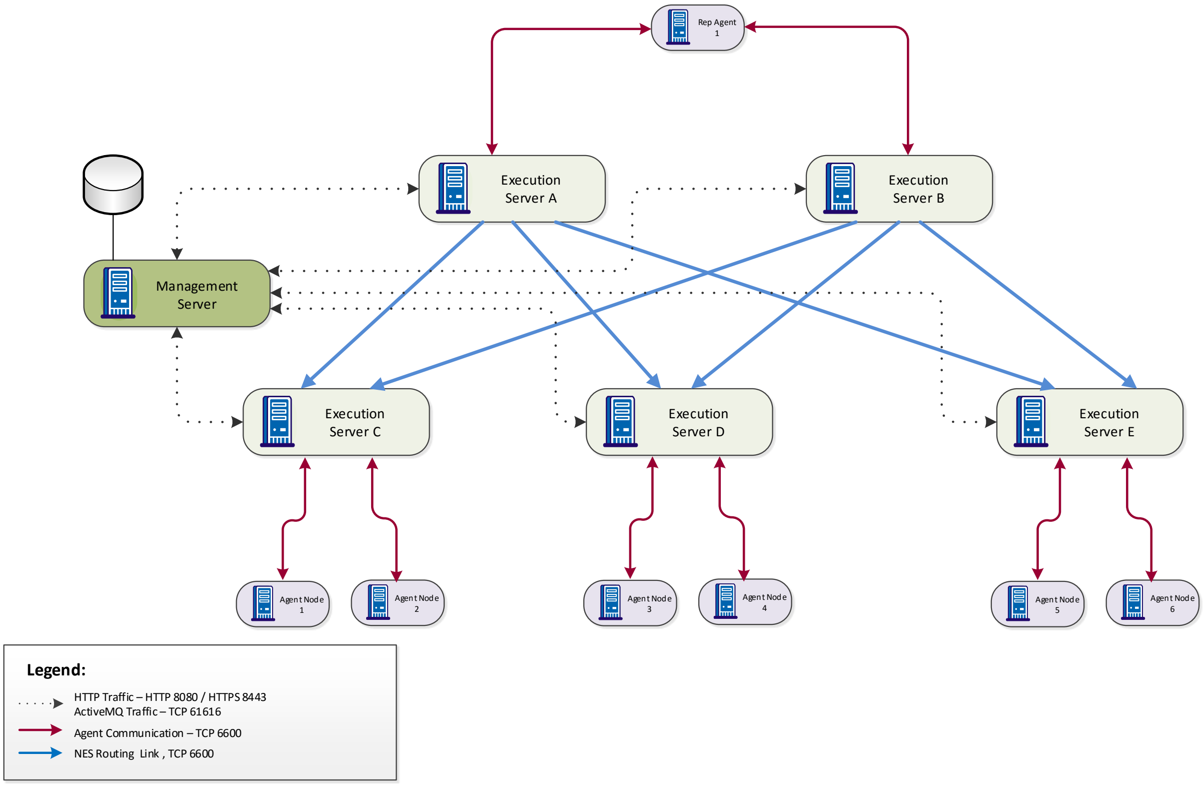
For small and medium size environments (Up to 4 Execution Servers and 2500 agents) the execution server ring architecture can be used. This creates a highly available Execution Server architecture and as you can see in the following schematic each data center contains two execution servers these are joined in a ring to two more execution servers in another center this ensures that there are two routes to each execution server. Agents In each data center are configured to connect to both datacentre execution servers in the event of an execution server failing the agents will automatically switch to the secondary execution server.

The blue lines indicate the execution server inter connections which are used to transfer artifacts between agents / execution servers. The best practice when creating this Execution Server routing architecture is to define the connection from the central execution server as shown by the arrows in the drawing, the connection is initiated from the source execution server but once the connection is made data flows in both directions.



Double STAR Execution Server with High Availability Architecture

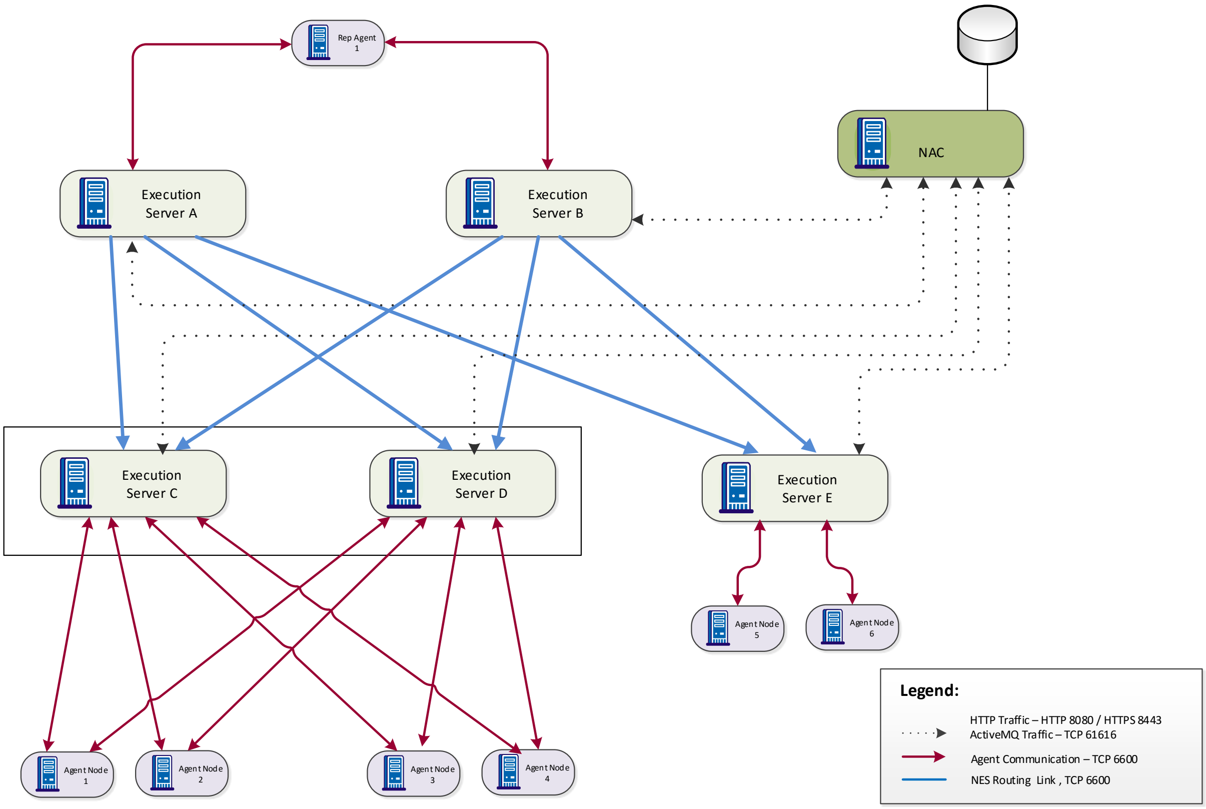
When dealing with a large environment (more than 4 Execution Servers and greater than 2500 Agents) it is recommended is to use a double STAR Execution Server routing architecture. A single STAR architecture introduces a single point of failure if “Execution Server A” is unavailable this will stop any inter Execution Server file transfers in this instance a STAR Execution Server with High Availability architecture can be used as per the following schematic, this architecture includes a secondary route between execution servers. The blue lines indicate the execution server inter connections which are used to transfer artifacts between agents / execution servers. The best practice when creating this Execution Server routing architecture is to define the connection from the central execution server as shown by the arrows in the drawing, the connection is initiated from the source execution server but once the connection is made data flows in both directions.



Note: Before configuring a STAR architecture ensure that the environment is upgraded to the latest maintenance build.

STAR Execution Server with High Availability Architecture and Super Execution Server Layer

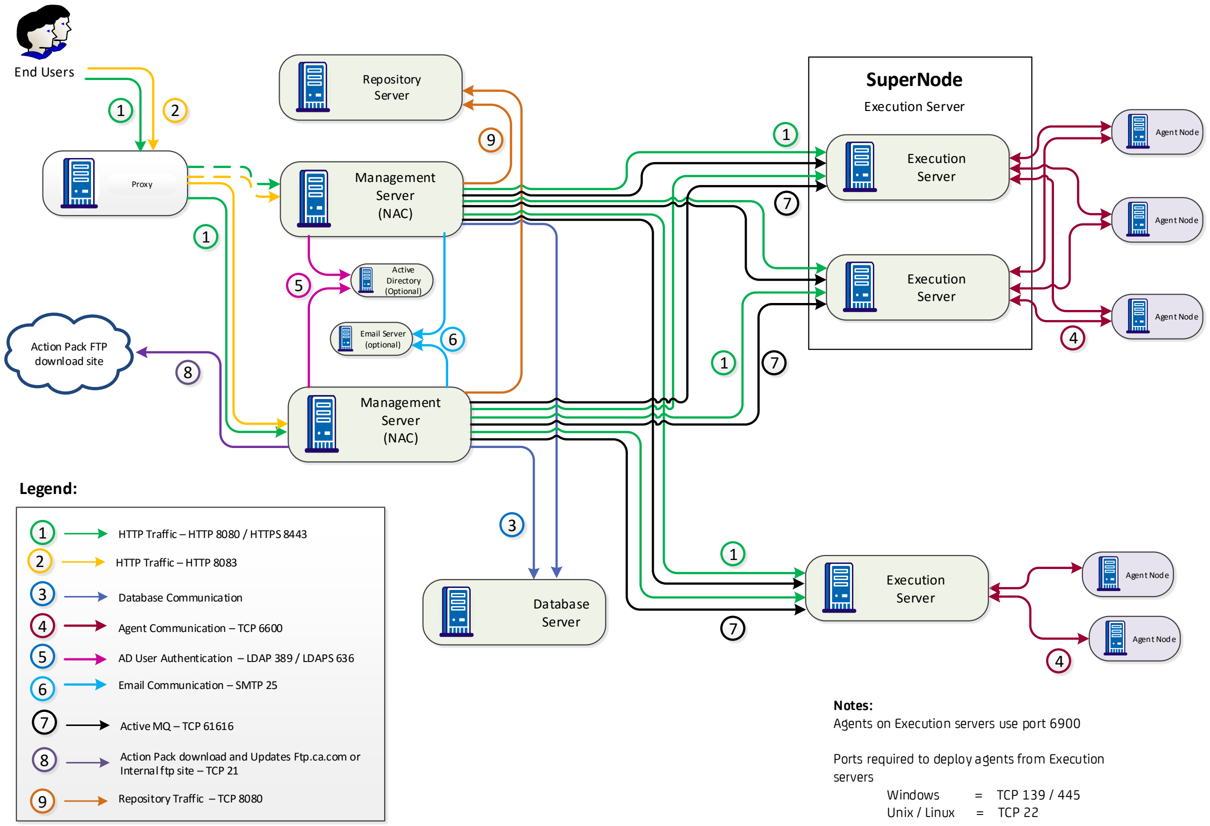
For complete fault tolerance Execution servers can be paired into what is called a “Super Execution Server” and the agents can be configured to communicate with both execution servers, in the event of an execution failing within a “Super Execution Server” the NAC will communicate with the agent via the remaining Execution Server. The blue lines indicate the execution server inter connections which are used to transfer artifacts between agents / execution servers. The best practice when creating this Execution Server routing architecture is to define the connection from the central execution server as shown by the arrows in the drawing, the connection is initiated from the source execution server but once the connection is made data flows in both directions.



Note: Before configuring a STAR architecture ensure that the environment is upgraded to the latest maintenance build.

Highly Available Architecture

The following section will explain how the CA Release Automation components can be deployed in a highly available environment and the steps required for configuring each of the components.



Legend:

1 → HTTP Traffic – HTTP 8080 / HTTPS 8443

2 → HTTP Traffic – HTTP 8083

3 → Database Communication

4 → Agent Communication – TCP 6600

5 → AD User Authentication – LDAP 389 / LDAPS 636

6 → Email Communication – SMTP 25

7 → Active MQ – TCP 61616

8 → Action Pack download and Updates Ftp.ca.com or Internal ftp site – TCP 21

9 → Repository Traffic – TCP 8080

Notes:

Agents on Execution servers use port 6900

Ports required to deploy agents from Execution servers

Windows	=	TCP 139 / 445
Unix / Linux	=	TCP 22

Management Server (NAC)

The Management Server component can be installed in a Highly Available (Active / Standby) configuration. The pre-requisites for installing the Active / Standby configuration are as follows:

1 Management Servers and proxy time must be synchronized to the second
2 Management Servers must be running on the same Operating System.
3 Management Servers and reverse proxy must be on the same LAN.

The steps to build this environment are as follows:

1 Install a Database server remotely from the Management Server and follow the steps in the administration guide to ensure the database is available for the installation of CA Release Automation.

2 Install a remote Nexus repository in a highly available configuration

3 To Install the Primary Management Server use the CA Release Automation custom installation do not install an Execution server, configure the Management Server with the details for the database server created in step 1 and connect to the highly available repository that was defined in step 2

4 On the Secondary Management Server use the CA Release Automation custom installation do not install an Execution server, configure the Management Server with the details for the same database server as the Primary node and connect to the highly available repository that was defined in step 2

5 A network Load Balancer is required to redirect network traffic to the Management Server Master node the Load Balancer functionality is not part of the CA solution. The Load balancer must be configured using a weighted priority algorithm with the Preferred NAC having a priority of 1.

6 The following URL can be used to confirm that the Release Automation server is running: http://*Managementserver:8080*/datamanagement/availability. When the Management server is running this URL a return code of 200 is returned

Database Server

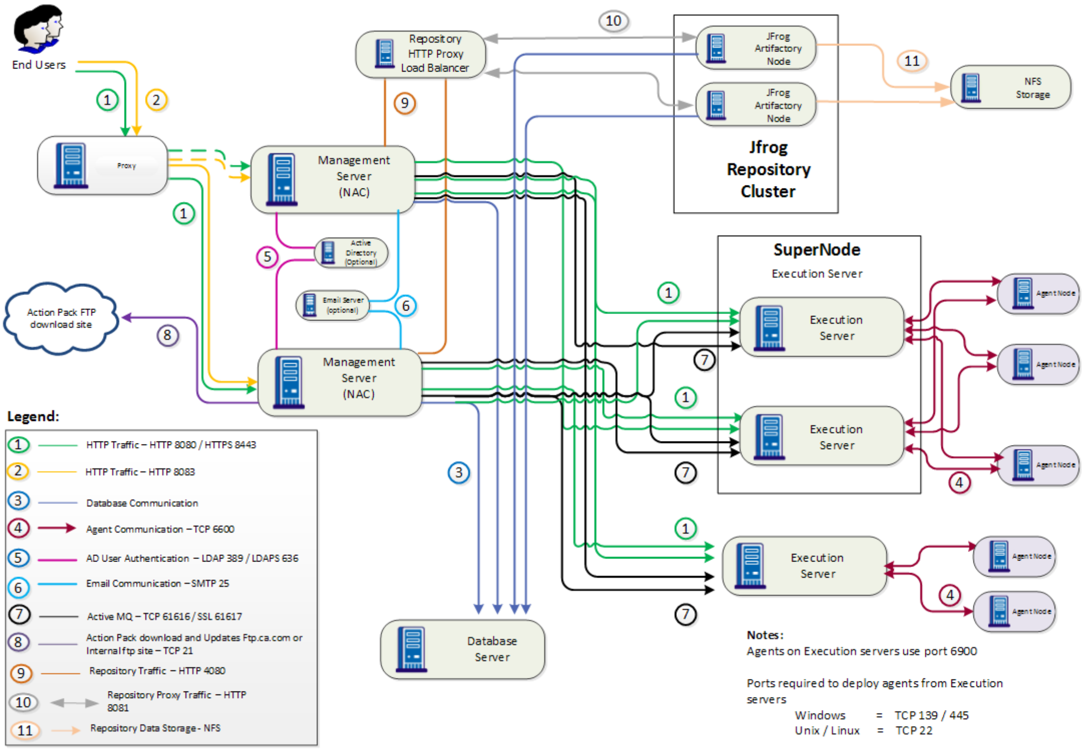
The Database Server should be highly resilient (Clustered) and be installed remotely from the Management Server

Repository Server - Nexus

When configuring the Management servers as highly available the repository server should be installed remote from the NAC and in a highly available configuration. The Nexus repository can be configured in an active passive configuration and in the event of a repository failure release automation can be reconfigured to redirect to the secondary node, details can be found on the Sonatype website for this configuration https://support.sonatype.com/entries/21451383-How-to-Set-up-Active-Standby-Failover.

Repository Server - jFrog

If the release automation repository is required to be configured in an active / active configuration then the NEXUS repository that is used internally within release automation can be replaced with the jFrog Artifactory repository which can be configured in a fully Highly Available architecture as shown in the following architecture drawing.


End Users
Proxy
Repository HTTP Proxy Load Balancer
JFrog Artifactory Node
JFrog Artifactory Node
Jfrog Repository Cluster
NFS Storage
Management Server (NAC)
Active Directory (Optional)
Email Server (optional)
Management Server (NAC)
Action Pack FTP download site
SuperNode
Execution Server
Execution Server
Execution Server
Execution Server
Agent Node
Agent Node
Agent Node
Agent Node
Agent Node
Database Server
1
2
3
4
5
6
7
8
9
10
11
Legend:
1 HTTP Traffic – HTTP 8080 / HTTPS 8443
2 HTTP Traffic – HTTP 8083
3 Database Communication
4 Agent Communication – TCP 6600
5 AD User Authentication – LDAP 389 / LDAPS 636
6 Email Communication – SMTP 25
7 Active MQ – TCP 61616 / SSL 61617
8 Action Pack download and Updates Ftp.ca.com or Internal ftp site – TCP 21
9 Repository Traffic – HTTP 4080
10 Repository Proxy Traffic – HTTP 8081
11 Repository Data Storage - NFS
Notes:
Agents on Execution servers use port 6900
Ports required to deploy agents from Execution servers
Windows = TCP 139 / 445
Unix / Linux = TCP 22

Details for this configuration can be found on the CA community site on this URL - https://communities.ca.com/docs/DOC-231150560

Execution Server

Execution server routing should be configured to allow for redundant routing and Execution servers should be deployed to create “Super Nodes” in this configuration agents are configured with a primary and secondary Execution server if the agent cannot communicate with the primary execution server it will switch to the secondary execution server.

During the agent installation the Primary execution server details are stored in the <Nolio Agent root install directory>/conf/nimi_config.xml under the supernodes section. In order to add another Execution Server you should edit nimi_config.xml and add another value to the supernodes section, the value will include the IP/Hostname of the secondary Execution Server and the port that the Execution Server needs to bind to communicate with the Agent. Below you can see an example how of the Agent is configured to work with two Execution Server

```
<supernodes>

<supernode>Execution-SRV1:6600</supernode>

<supernode>Execution-SRV2:6600</supernode>

</supernodes>
```

An alternative method to configure the execution servers that an agent is registered with is to select the agent / agents in the ASAP UI administration tab / Agent Management and Right Click on the agents selecting the option “Change Execution Server of the selected agents” all execution servers will be displayed if you select the execution servers that the agent should be registered with and click OK the agents will be reconfigured

Firewall Architecture Scenarios

The following section includes some examples of how Release Automation can be configured in a firewalled environment.

Scenario One – Agent outside firewall and only outbound traffic allowed

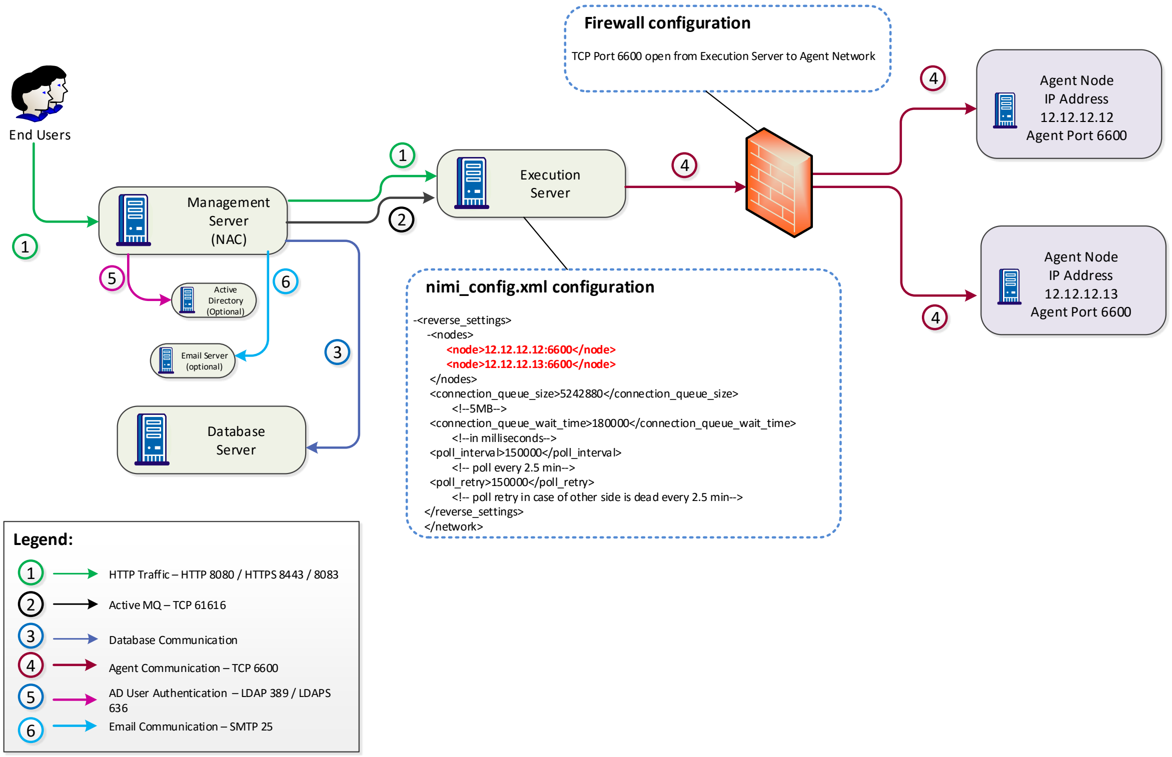
In the first scenario the servers to be managed with Release Automation are located outside of a firewall in a DMZ and network traffic is only allowed to travel from the internal network to the DMZ. Normally a Release Automation agent will connect to the Execution Server but in this scenario the connection must be initiated from the Execution Server these are the high level deployment steps:

1 The agent should be installed on the servers in the DMZ using the silent installer. The silent installer includes a switch that will override the default setting to validate the connection from the agent to the Execution Server, for further details on this please refer to the CA SWAT guide – Zero touch deployment

2 Configure the firewall to allow traffic from the internal network to the DMZ using TCP port 6600

3 Configure the reverse settings section of the nimi_config.xml on the Execution Server with the IP addresses of the agent outside the firewall.

4 Restart the Execution Server service, the Execution Server will periodically (2.5 Minute Default) attempt to connect to the agent and if the agent responds the session will be kept open through the firewall.


Firewall configuration
TCP Port 6600 open from Execution Server to Agent Network
End Users
1
Management Server (NAC)
1
2
Execution Server
4
4
Agent Node IP Address 12.12.12.12 Agent Port 6600
4
Agent Node IP Address 12.12.12.13 Agent Port 6600
5
Active Directory (Optional)
6
Email Server (optional)
3
Database Server
nimi_config.xml configuration

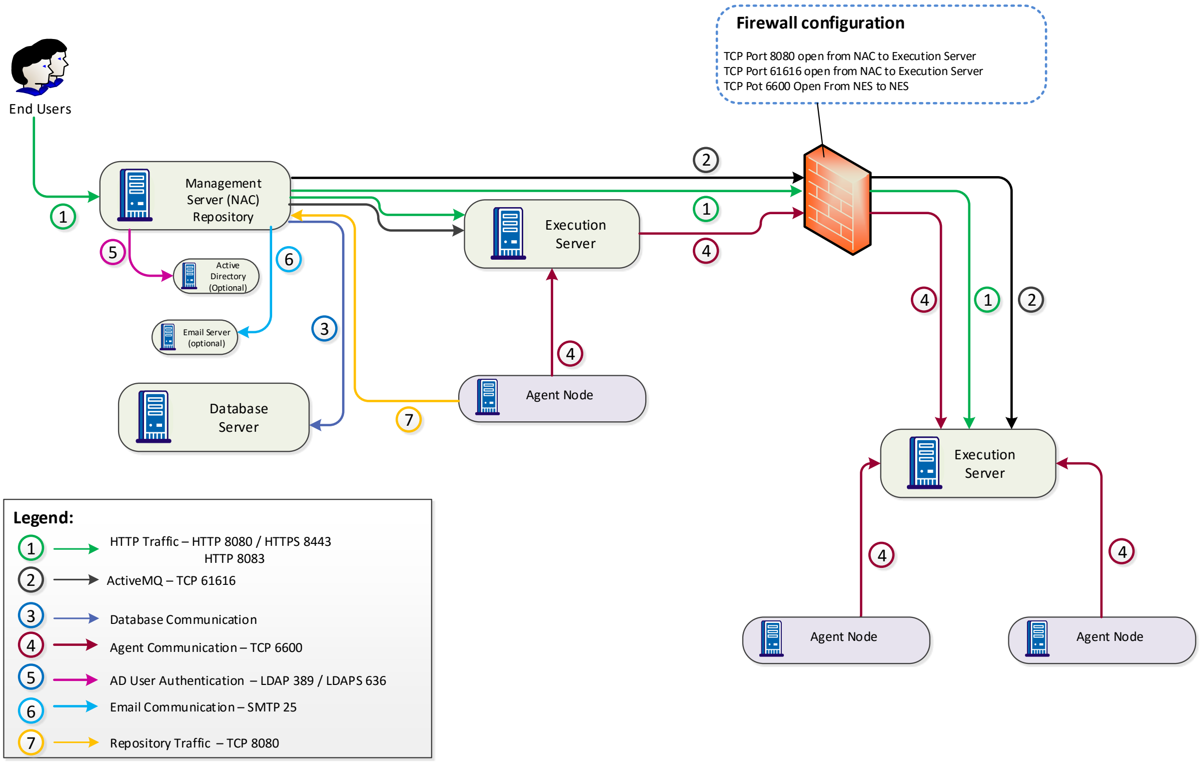
```
-<reverse_settings>
  -<nodes>
      <node>12.12.12.12:6600</node>
      <node>12.12.12.13:6600</node>
   </nodes>
   <connection_queue_size>5242880</connection_queue_size>
        <!--5MB-->
   <connection_queue_wait_time>180000</connection_queue_wait_time>
        <!--in milliseconds-->
   <poll_interval>150000</poll_interval>
        <!-- poll every 2.5 min-->
   <poll_retry>150000</poll_retry>
        <!-- poll retry in case of other side is dead every 2.5 min-->
  </reverse_settings>
  </network>
```

Legend:

1	HTTP Traffic – HTTP 8080 / HTTPS 8443 / 8083
2	Active MQ – TCP 61616
3	Database Communication
4	Agent Communication – TCP 6600
5	AD User Authentication – LDAP 389 / LDAPS 636
6	Email Communication – SMTP 25

Scenario Two – Execution Server placed outside Firewall

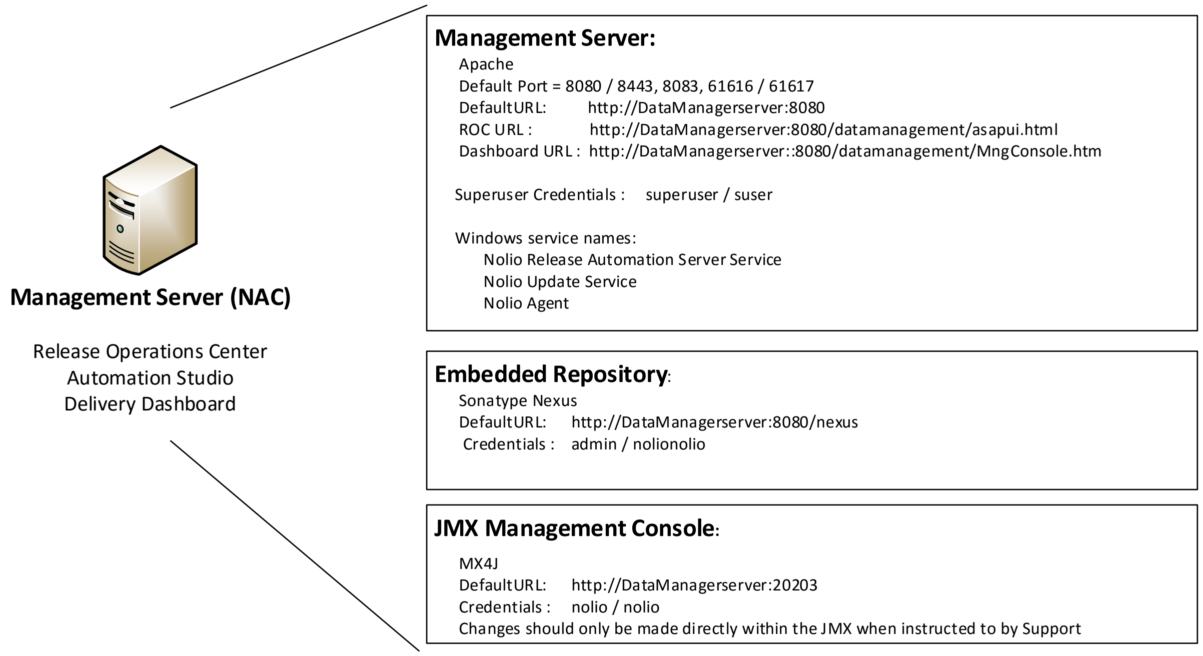
In the second scenario the servers to be managed with Release Automation are located outside of a firewall in a DMZ and network traffic, the agent inside the firewall is acting as a repository agent and is connected to an Execution Server inside the firewall. Execution Server routing has been defined between the two Execution Servers


Firewall configuration
TCP Port 8080 open from NAC to Execution Server
TCP Port 61616 open from NAC to Execution Server
TCP Pot 6600 Open From NES to NES
End Users
Management Server (NAC) Repository
Execution Server
Active Directory (Optional)
Email Server (optional)
Database Server
Agent Node
Execution Server
Agent Node
Agent Node
1
2
3
4
5
6
7
Legend:
1 HTTP Traffic – HTTP 8080 / HTTPS 8443 HTTP 8083
2 ActiveMQ – TCP 61616
3 Database Communication
4 Agent Communication – TCP 6600
5 AD User Authentication – LDAP 389 / LDAPS 636
6 Email Communication – SMTP 25
7 Repository Traffic – TCP 8080

Appendix A

Management Server - URL's / Default Credentials and Logs

The following graphic shows the various internal component's that make up a Management Server with the corresponding connection details.



Management Server:
Apache
Default Port = 8080 / 8443, 8083, 61616 / 61617
DefaultURL: http://DataManagerserver:8080
ROC URL : http://DataManagerserver:8080/datamanagement/asapui.html
Dashboard URL : http://DataManagerserver::8080/datamanagement/MngConsole.htm

Superuser Credentials : superuser / suser

Windows service names:
Nolio Release Automation Server Service
Nolio Update Service
Nolio Agent

Embedded Repository:
Sonatype Nexus
DefaultURL: http://DataManagerserver:8080/nexus
Credentials : admin / nolionolio

JMX Management Console:
MX4J
DefaultURL: http://DataManagerserver:20203
Credentials : nolio / nolio
Changes should only be made directly within the JMX when instructed to by Support

Management Server Log files overview:

1. nolio_dm_all.log
   a. Log file contains the following information:
      i. NAC start-up sequence.
      ii. DB connectivity
      iii. Amount of agents that connected to each Execution Server and unreachable agents.
      iv. Status of processes execution
      v. Logged in users
      vi. Details about design and publish activities
   b. File Location = <Install dir>/logs/nolio_dm_all.log
2. Action Pack Update Service Logging
   a. Contains information about the action pack update service., the update-service.log is the primary log and contains information on the action pack download status
   b. File Location = <Install dir>\UpdateService
   c. nolio_update_service_error.log
   d. nolio_update_service_output.log
   e. update-service.log
3. ActiveMQ Logging
   a. Contains information about ActiveMQ connectivity
   b. File Location = <Install dir>/logs/active_mq_nac.log
4. nolio_document.log

    a. Contains information about processes that exported to xml document
    b. File Location = <Install dir>/logs/nolio_document.log

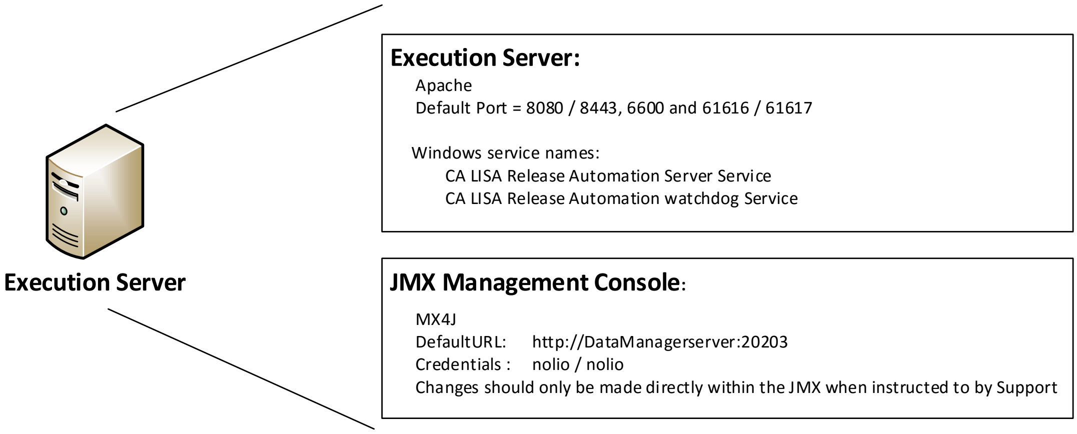
5. nolio_export.log
    a. Contains information about components/applications that imported/exported to/from the system
    b. LogFile Location = <Install dir>/logs/nolio_export.log
6. nolio_auditing.log
    a. Contains all design and administration changes (Note that audit report need to be enable)
    b. Log file Location = <Install dir>/logs/nolio_auditing.log
7. installation.log
    a. Contains a summary of system installation
    b. Log file Location = <Install dir>/.install4j/installation.log
8. installation.log.*
    a. Contains a summary of system upgrade from previous version
    b. <Install dir>/.install4j/installation.log.*

9. Agent_upgrade.log
    a. Contains a summary of agents upgrade
    b. Log file Location = <Install dir>/logs/Agent_upgrade.log
10. Installation log can be found in %temp% folder

Execution Server (NES) Internals - URL's / Default Credentials and Logs

The execution server is the link between the NAC / Management server and the Agent Nodes, requests from the NAC / Management Server to agents are sent via the Execution server. Additional execution servers can be added when working with complex networks, remote data centers or to scale out the solution. An execution server by default is configured to support 200 Agents, this figure could be higher if the workload for the agents is particularly low.

The following graphic shows the various internal component's that make up an Execution Server with the corresponding connection details.



Execution Server:

Apache
Default Port = 8080 / 8443, 6600 and 61616 / 61617

Windows service names:

CA LISA Release Automation Server Service
CA LISA Release Automation watchdog Service

JMX Management Console:

MX4J
DefaultURL: http://DataManagerserver:20203
Credentials : nolio / nolio
Changes should only be made directly within the JMX when instructed to by Support

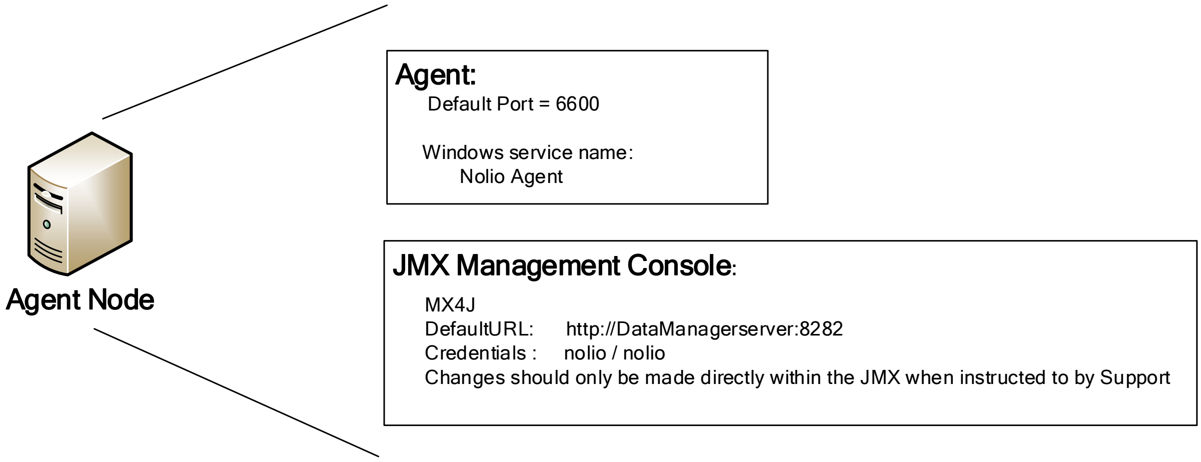
Execution Server Log files description:

1. Nimi.log
   a. Contains Information regarding communication between NAGs and Execution Server such as handshake activity
   b. Contains Network topology (NAG and Execution Server versions, ID's IP's etc.)
   c. Contains Information regarding parameters values and files that transfers between NAGs.
   d. Log file location = <Install dir>/logs/nimi.log
2. Nolio_exec_all.log and execution.log
   a. Contains Information regarding execution events and parameters that transfers between NAG's Execution Server and NAC.
   b. Contains Remote agent installations logging
   c. Log file Locations = <Install dir>/logs/Nolio_exec_all.log, <Install dir>/logs/execution.log
3. ActiveMQ Logging
   a. Contains information about ActiveMQ connectivity
   b. File Location = <Install dir>/logs/active_mq_nes.log
4. installation.log
   a. Contains a summary of system installation
   b. Log file Location = <Install dir>/.install4j/installation.log
5. installation.log.*
   a. Contains a summary of system upgrade from previous version
   b. <Install dir>/.install4j/installation.log.*
6. Installation log can be found in %temp% folder

Agent Node (AGT) – Default Ports and Logs

The agent node communicates with the execution server and utilises a proprietary communication protocol called nimi.

The following graphic shows the core component's that make up an Agent node



Agent:
Default Port = 6600

Windows service name:
Nolio Agent

JMX Management Console:
MX4J
DefaultURL: http://DataManagerserver:8282
Credentials : nolio / nolio
Changes should only be made directly within the JMX when instructed to by Support

Agent node Log files description:

1. Nimi.log
    a. Contains Information regarding communication between agent and Execution Server
    b. Contains Information regarding parameters values and files that transfers
    c. Log file location = <Install dir>/logs/nimi.log
2. Nolio_all.log
    a. All NAG activity except the network layer (stored in nimi.log)
    b. Log file Locations = <Install dir>/logs/Nolio_all.log
3. Nolio_action_exe.log
    a. Contains specific information about actions executions and their results.
    b. Log file Locations = <Install dir>/logs/
4. installation.log
    a. Contains a summary of system installation
    b. Log file Location = <Install dir>/install4j/installation.log
5. Installation log can be found in %temp% folder

Appendix B - Tuning tips

The following section contains details on modifying default configuration settings and any updates should only be made after consulting with CA support / Engineering

Execution Server Tuning

Communication between execution servers and agents is via the protocol nimi which is configured in the \CA\ReleaseAutomationServer\conf\nimi_config.xml. Two important settings are:

1. The Capacity element setting is used to configure the maximum number of agents that can be registered to the Execution Server and the default value is 200. When 200 agents are registered to the execution server any further registration requests will be rejected.
2. The Warn-capacity.element setting should always be lower than the Capacity element setting. When the number of agent registered with the execution server exceeds this value the new connecting nodes will be asked to seek another supernode

When the execution server is handling a large number of agents or is configured with complex routing such as STAR and Double STAR the execution servers should be configured as follows and the latest maintenance must be applied. The following settings should also be applied to any retrieval agents.

Add a new file **ra_services.properties** at the conf folder of the Execution Server servers and the Retrieval agent.

The file conf/ra_services.properties should include the following:

```
==================================================================

CommunicationNetworkServiceManager.NumberThreads=30

CommunicationNetworkServiceManager.QueueSize=300

==================================================================
```

Upon server restart, you should see the following lines in the server log file nolio_exec_all.log:

```
2014-10-21 09:31:10,855 [nesTaskScheduler-1] DEBUG
(com.nolio.platform.shared.communication.CommunicationNetworkConfiguration:51) -
communication.properties: {PostOffice.class=com.nolio.nimi.NimiPostOffice}

...

2014-10-21 09:34:13,468 [New I/O server worker #1-1]
INFO  (com.nolio.platform.shared.ServicesConfiguration:108) - The
[CommunicationNetworkServiceManager.NumberThreads] was found with value:30
```

```
2014-10-21 09:34:13,468 [New I/O server worker #1-1]
INFO  (com.nolio.platform.shared.ServicesConfiguration:108) - The
[CommunicationNetworkServiceManager.QueueSize] was found with value:300
```

The nimi_config.xml should be updated as follows:

Parameter	Description	Default Setting	New Setting
appmessages > threadpool > size	This parameter controls the number of threads pools. Increasing the number of threads allows the Execution Server/Retrieval Agent to handle a larger rate of incoming/outgoing messages (events/messages per second).	30	60
network > threadpool > client > size	This parameter controls the number of threads pools. Increasing the number of threads allows the Execution Server/Retrieval Agent to handle a larger rate of incoming/outgoing messages (events/messages per second).	2	4
network > threadpool > server > size	This parameter controls the number of threads pools. Increasing the number of threads allows the Execution Server/Retrieval Agent to handle a larger rate of incoming/outgoing messages (events/messages per second).	2	4
network > threadpool > outbound > size	This parameter controls the number of threads pools. Increasing the number of threads allows the Execution Server/Retrieval Agent to handle a larger rate of incoming/outgoing messages (events/messages per second).	30	60
network > threadpool > reverse > size	This parameter controls the number of threads pools. Increasing the number of threads allows the Execution Server/Retrieval Agent to handle a larger rate of incoming/outgoing messages (events/messages per second).	30	60
network > bad_destination_throttle_time	Decreasing this value will reduce the duration of the recovery period after receiving “bad route” response	90000	15000
network > timeout > idle	Increasing from 120 sec to 180 sec – would keep an open connection between agent and Execution Server for 3 minutes (even if it is idle).	120000	180000

	This should allow the Execution Server to immediately deliver messages to agents without delay.		
routing > threadpool > size		30	60
routing > timeout > request		180000	180000
routing > retry > count	Decreasing this value would prevent redundant unnecessary retries of route requests. In case, there is a route from one agent to another – it should not take more than 15 sec to detect it.	3	0
routing > retry > duration	Decreasing this value would prevent redundant unnecessary retries of route requests. In case, there is a route from one agent to another – it should not take more than 15 sec to detect it.	60000	15000
keepalive > threadpool > size		30	60
files > threadpool > size		30	60

These settings should be applied across all of the Execution Servers starting with the central Execution Server servers, then updating the datacenter Execution Server servers and finally update the retrieval agents.

Non retrieval agents do not need to be updated with these settings and should keep the default standard **nimi_config.xml** setup.

Modifying default ports

Modify default Tomcat server port.

The port used for the ROC and the ASAP is configured in the following configuration file:

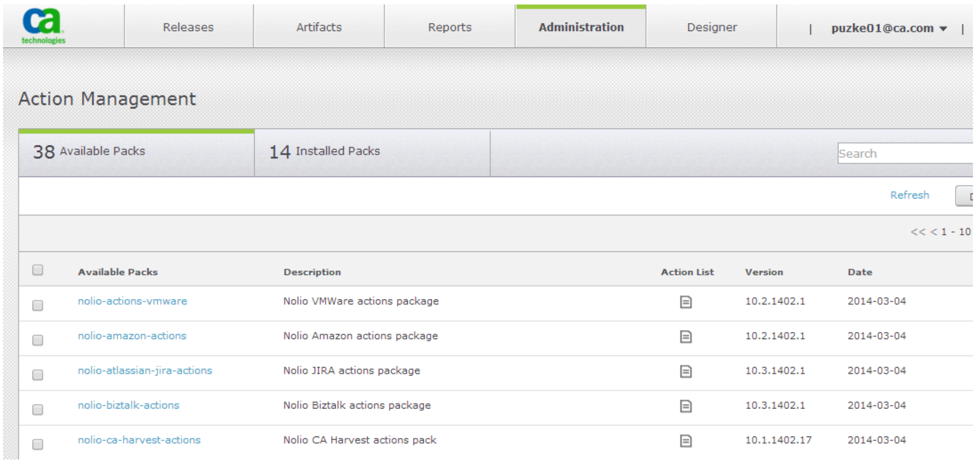
"C:\Program Files\CA\ReleaseAutomationServer\conf\server.xml"

The port is configured in the Connection section as shown below, after modifying the configuration file the Release automation service should be restarted.

```
<Connector connectionTimeout="20000" port="8080"
protocol="org.apache.coyote.http11.Http11NioProtocol" redirectPort="8443"/>
```

Action Pack download service Configuration

The Action Management panel in the Release Operations Center displays the installed Action Packs and also a list of available action packs that can be downloaded from the CA FTP site as shown in the following screenshot.



Action Management

38 Available Packs | 14 Installed Packs

□	Available Packs	Description	Action List	Version	Date
□	nolio-actions-vmware	Nolio VMWare actions package	🗎	10.2.1402.1	2014-03-04
□	nolio-amazon-actions	Nolio Amazon actions package	🗎	10.2.1402.1	2014-03-04
□	nolio-atlassian-jira-actions	Nolio JIRA actions package	🗎	10.3.1402.1	2014-03-04
□	nolio-biztalk-actions	Nolio Biztalk actions package	🗎	10.3.1402.1	2014-03-04
□	nolio-ca-harvest-actions	Nolio CA Harvest actions pack	🗎	10.1.1402.17	2014-03-04

The default ports for this service can be modified in the following file which can be found on the NAC:

"C:\Program Files\CA\ReleaseAutomationServer\UpdateService\URL.ini"

The configuration file contains the following information:

```
#Wed Jan 29 15:38:12 EST 2014

CA_URL=ftp\://ftp.ca.com/pub/dpm/ReleaseAutomation/UpdateService

CA_PACK_URL=ftp\://ftp.ca.com/pub/dpm/ReleaseAutomationActions/ActionPacks

INTERVAL=1440

REST_PORT=8083
```

The REST_PORT value is the port used to connect the Action management portlet within the ROC UI to the update service on the NAC.

Logs for the update service are stored in the following location <Install dir>\UpdateService . . The update-service.log contains all of the information from the update service and should be the first log to review when troubleshooting download issue, the following two logs contain additional logging for the service nolio_update_service_error.log, nolio_update_service_output.log

Modify Action pack FTP location

If the NAC does not have access to the internet the content of the CA FTP site can be copied to a FTP server on the NAC or another FTP server to achieve this copy the files from these two location to the new ftp server:

ftp.ca.com/pub/dpm/ReleaseAutomation/UpdateService

ftp.ca.com/pub/dpm/ReleaseAutomationActions/ActionPacks

Edit the URL .ini and modify the values for CA_URL and CA_PACK_URL to the ftp location of the ftp server. In the following example of the URL.ini the ftp server has been installed on the NAC and two folders created UpdateService and ActionPacks that contain the files from the CA FTP server .

```
#Wed Feb 18 15:38:12 GMT 2015

CA_URL=ftp\://localhost/UpdateService

CA_PACK_URL=ftp\://localhost/ActionPacks

INTERVAL=1440

REST_PORT=8083
```

After modifying the URL.ini on Windows restart the “Nolio Update Service” on Linux run the following command “service nolio_update_service restart”

ActiveMQ Configuration

The ActiveMQ port is set by default during the installation to TCP 61616 this port can be changed after the installation by modifying the file jms.properties

On the NAC the file location is

<RA Root>/webapps/<datamanagement>

And on the execution servers:

<RA Root>/webapps/<execution>/WEB-INF

To change the port modify the value "jms.transport.port=61616" to a new unused port and once the port has been changed the NAC and all Execution servers should be restarted.